

Spring Core_Maven

Scenario:

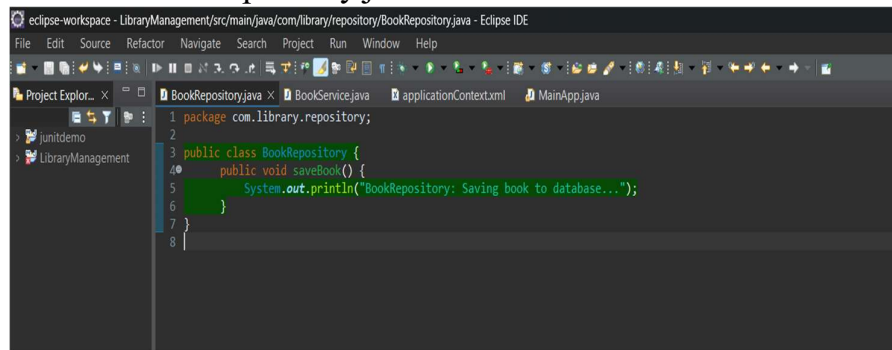
Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

Steps:

1. Set Up a Spring Project:
 - Create a Maven project named LibraryManagement.
 - Add Spring Core dependencies in the pom.xml file.
2. Configure the Application Context:
 - Create an XML configuration file named applicationContext.xml in the src/main/resources directory.
 - Define beans for BookService and BookRepository in the XML file.
3. Define Service and Repository Classes:
 - Create a package com.library.service and add a class BookService.
 - Create a package com.library.repository and add a class BookRepository.
4. Run the Application:
 - Create a main class to load the Spring context and test the configuration.

Solution:

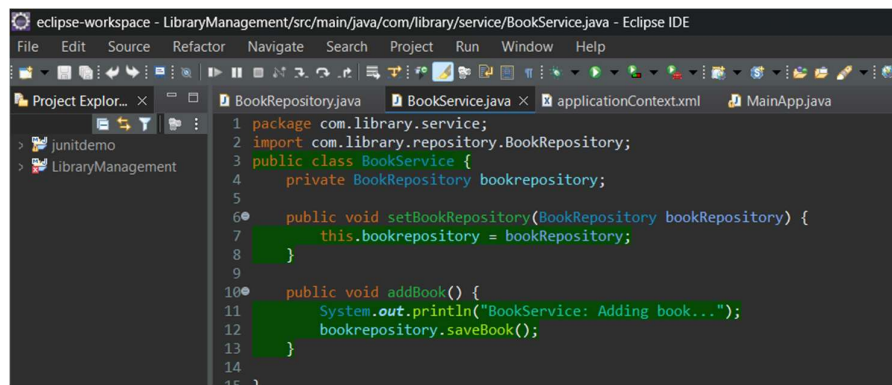
Created BookRepository.java



The screenshot shows the Eclipse IDE with the 'BookRepository.java' file open. The code defines a package 'com.library.repository' and a public class 'BookRepository' with a 'saveBook()' method that prints a message to the console.

```
1 package com.library.repository;
2
3 public class BookRepository {
4     public void saveBook() {
5         System.out.println("BookRepository: Saving book to database...");
6     }
7 }
8
```

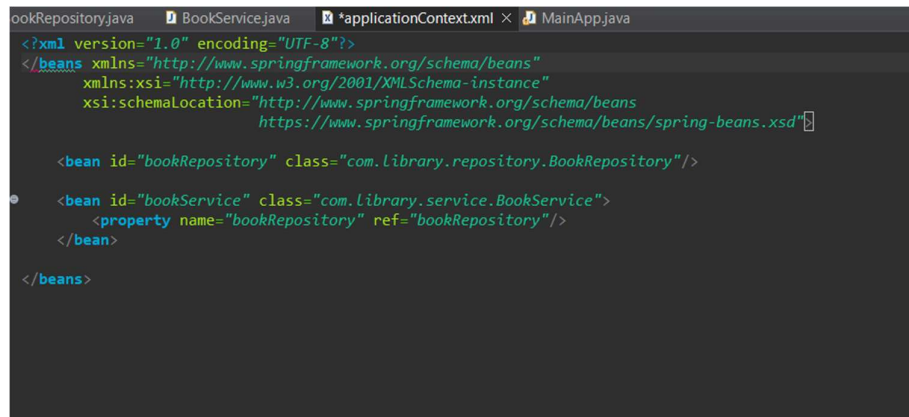
Created BookService.java



The screenshot shows the Eclipse IDE with the 'BookService.java' file open. The code defines a package 'com.library.service', imports 'BookRepository', and defines a 'BookService' class. It includes a 'setBookRepository()' method to inject the repository and an 'addBook()' method that uses the repository to save a book.

```
1 package com.library.service;
2 import com.library.repository.BookRepository;
3 public class BookService {
4     private BookRepository bookRepository;
5
6     public void setBookRepository(BookRepository bookRepository) {
7         this.bookRepository = bookRepository;
8     }
9
10    public void addBook() {
11        System.out.println("BookService: Adding book...");
12        bookRepository.saveBook();
13    }
14
15 }
```

Created applicationContext.xml



The screenshot shows the Eclipse IDE with the 'applicationContext.xml' file open. The XML file configures the Spring container with two beans: 'bookRepository' of type 'com.library.repository.BookRepository' and 'bookService' of type 'com.library.service.BookService'. The 'bookService' bean has a property 'bookRepository' that references the 'bookRepository' bean.

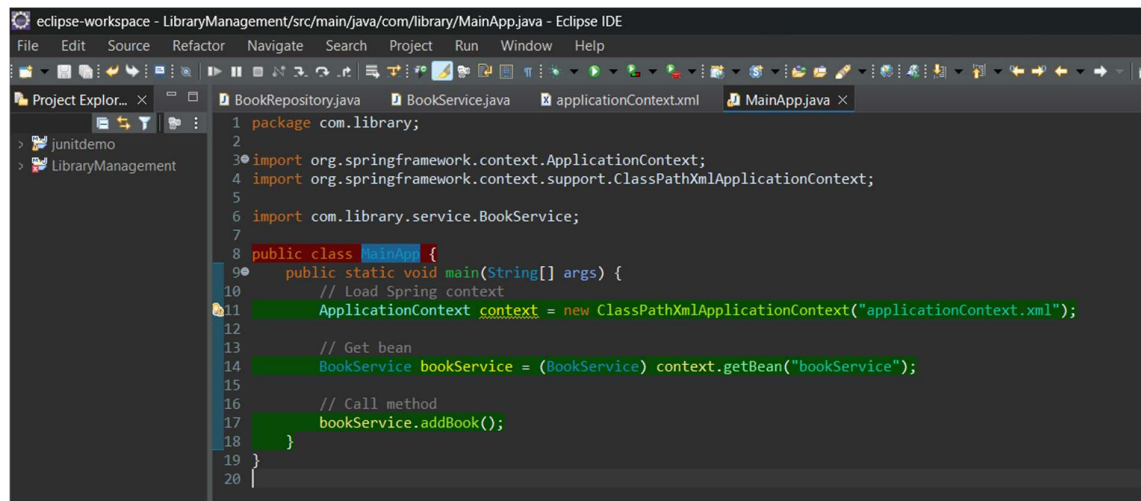
```
<?xml version="1.0" encoding="UTF-8"?>
</beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.library.repository.BookRepository"/>

    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

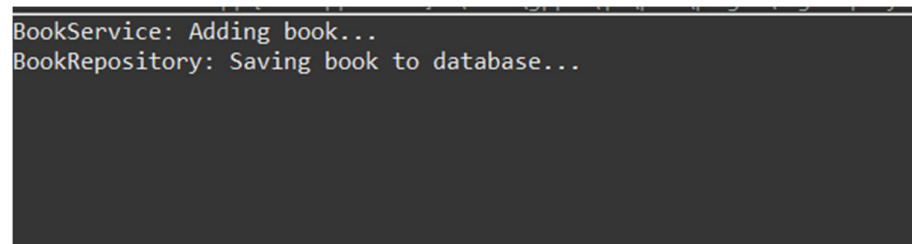
</beans>
```

Created the test class, MainApp.java

A screenshot of the Eclipse IDE interface. The title bar reads 'eclipse-workspace - LibraryManagement/src/main/java/com/library/MainApp.java - Eclipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development. The Project Explorer on the left shows a project named 'LibraryManagement'. The main editor area displays the code for 'MainApp.java'. The code is as follows:

```
1 package com.library;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 import com.library.service.BookService;
7
8 public class MainApp {
9     public static void main(String[] args) {
10         // Load Spring context
11         ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
12
13         // Get bean
14         BookService bookService = (BookService) context.getBean("bookService");
15
16         // Call method
17         bookService.addBook();
18     }
19 }
20
```

Output:

A screenshot of the output console in the IDE. It shows two lines of text: 'BookService: Adding book...' and 'BookRepository: Saving book to database...'.

```
BookService: Adding book...
BookRepository: Saving book to database...
```